OIPE
JAN 07 2005
PATENT & TRADEMARK OFFICE

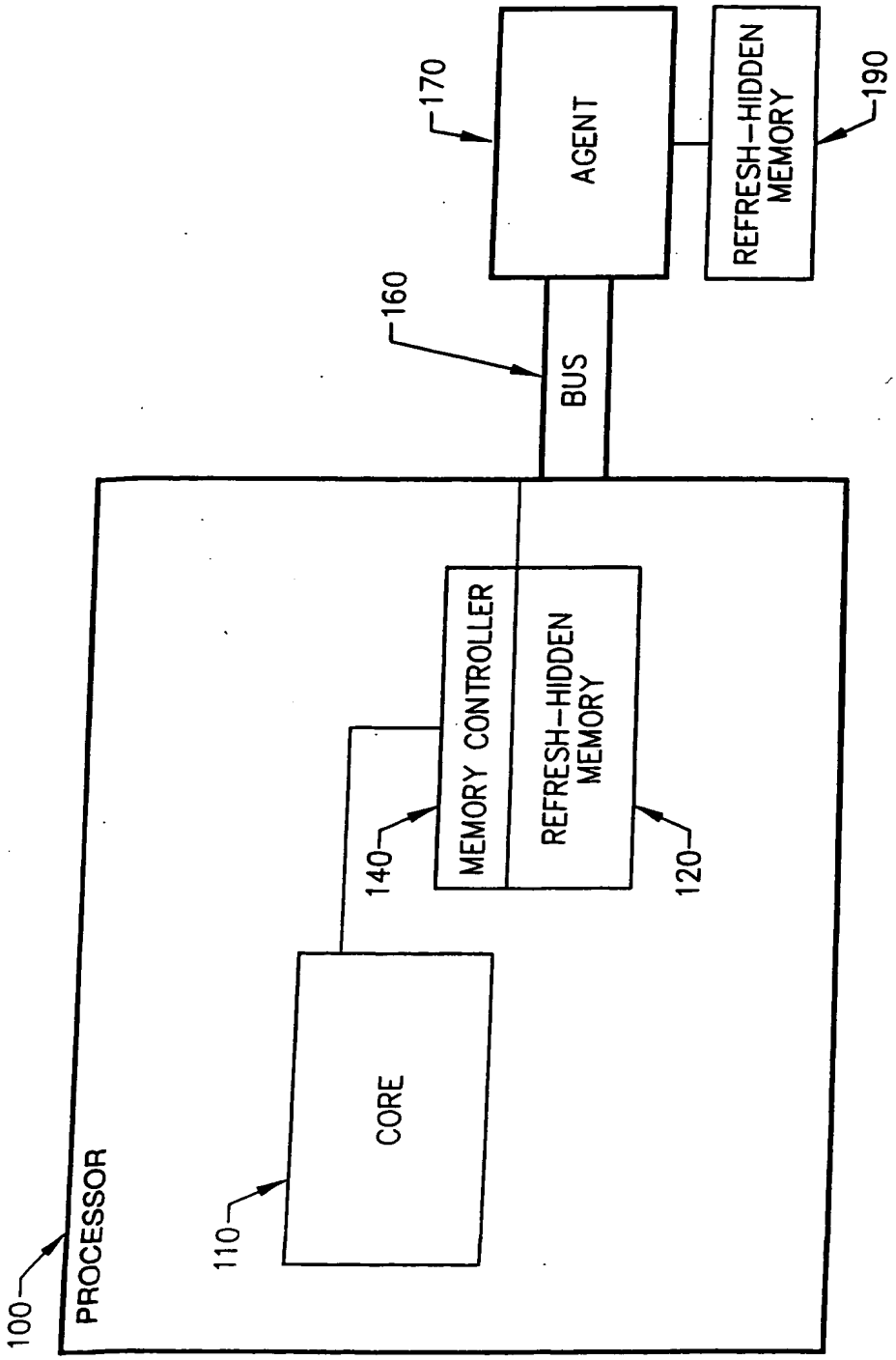


FIGURE 1

FIGURE 2

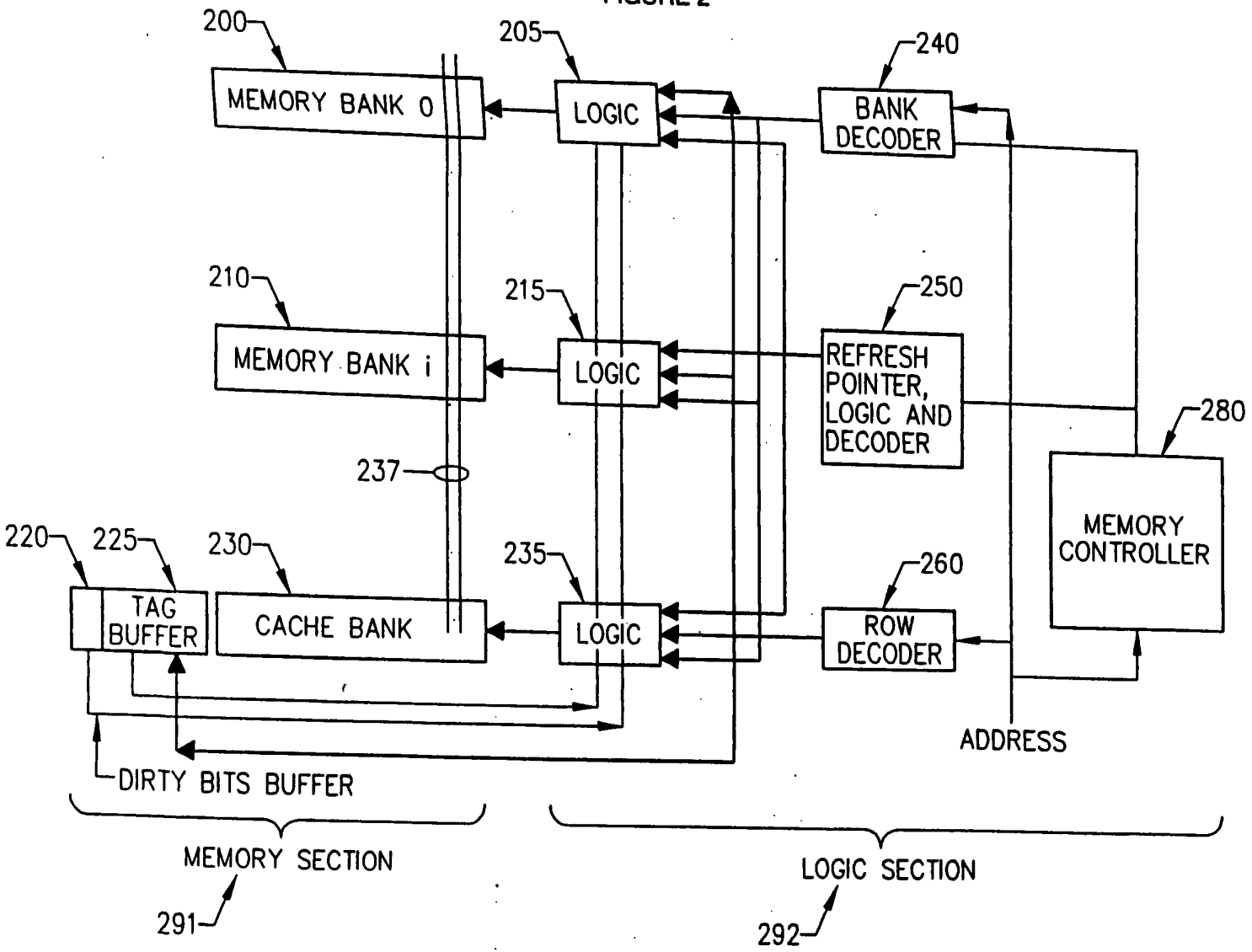

200
205
240
MEMORY BANK 0
LOGIC
BANK DECODER
210
215
250
MEMORY BANK i
LOGIC
REFRESH POINTER, LOGIC AND DECODER
280
237
MEMORY CONTROLLER
220
225
230
235
260
TAG BUFFER
CACHE BANK
LOGIC
ROW DECODER
ADDRESS
DIRTY BITS BUFFER
MEMORY SECTION
LOGIC SECTION
291
292

FIGURE 3

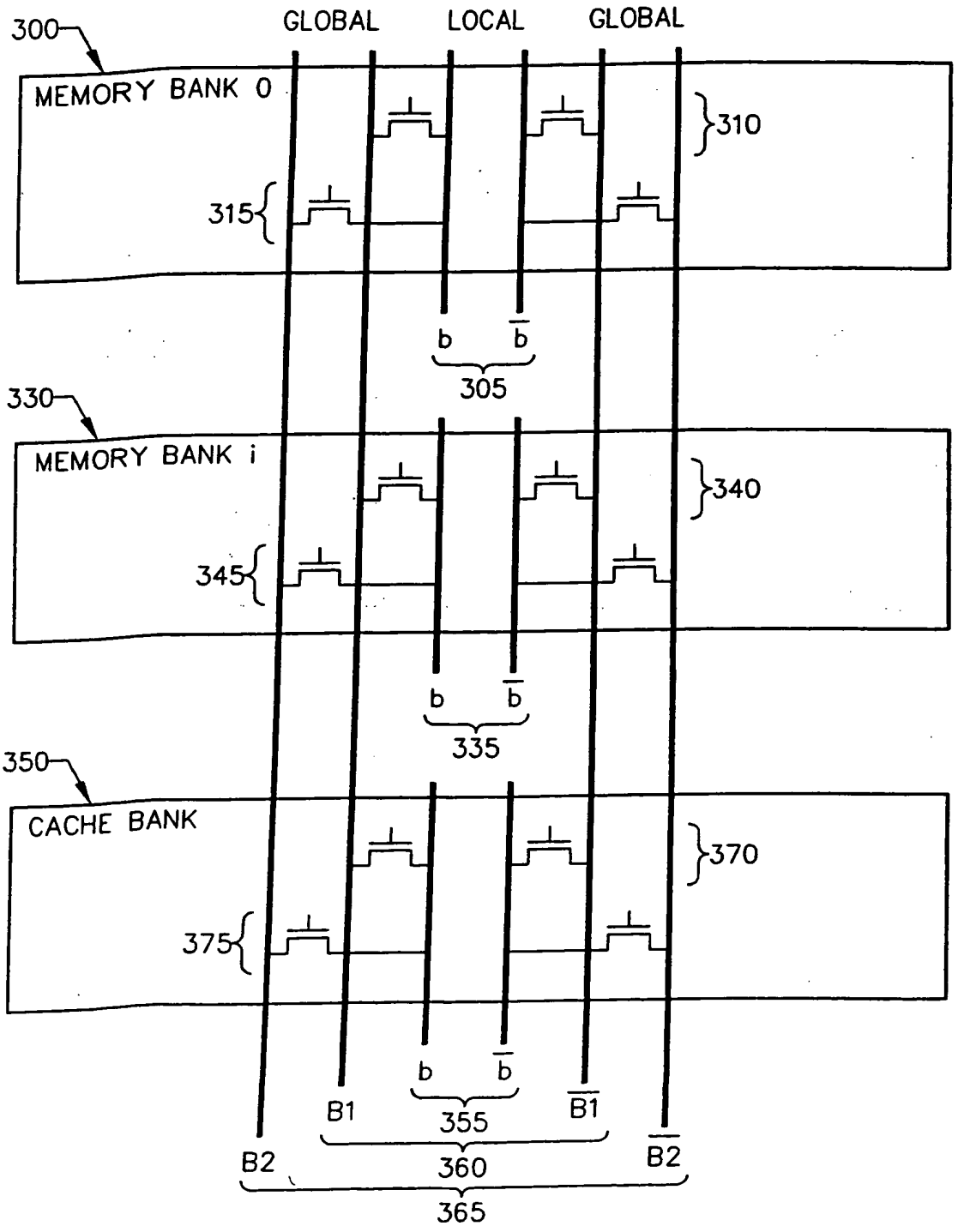

300
GLOBAL
LOCAL
GLOBAL
MEMORY BANK 0
310
315
b
$\overline{b}$
305
330
MEMORY BANK i
340
345
b
$\overline{b}$
335
350
CACHE BANK
370
375
b
$\overline{b}$
B1
355
$\overline{B1}$
B2
360
$\overline{B2}$
365

FIGURE 4

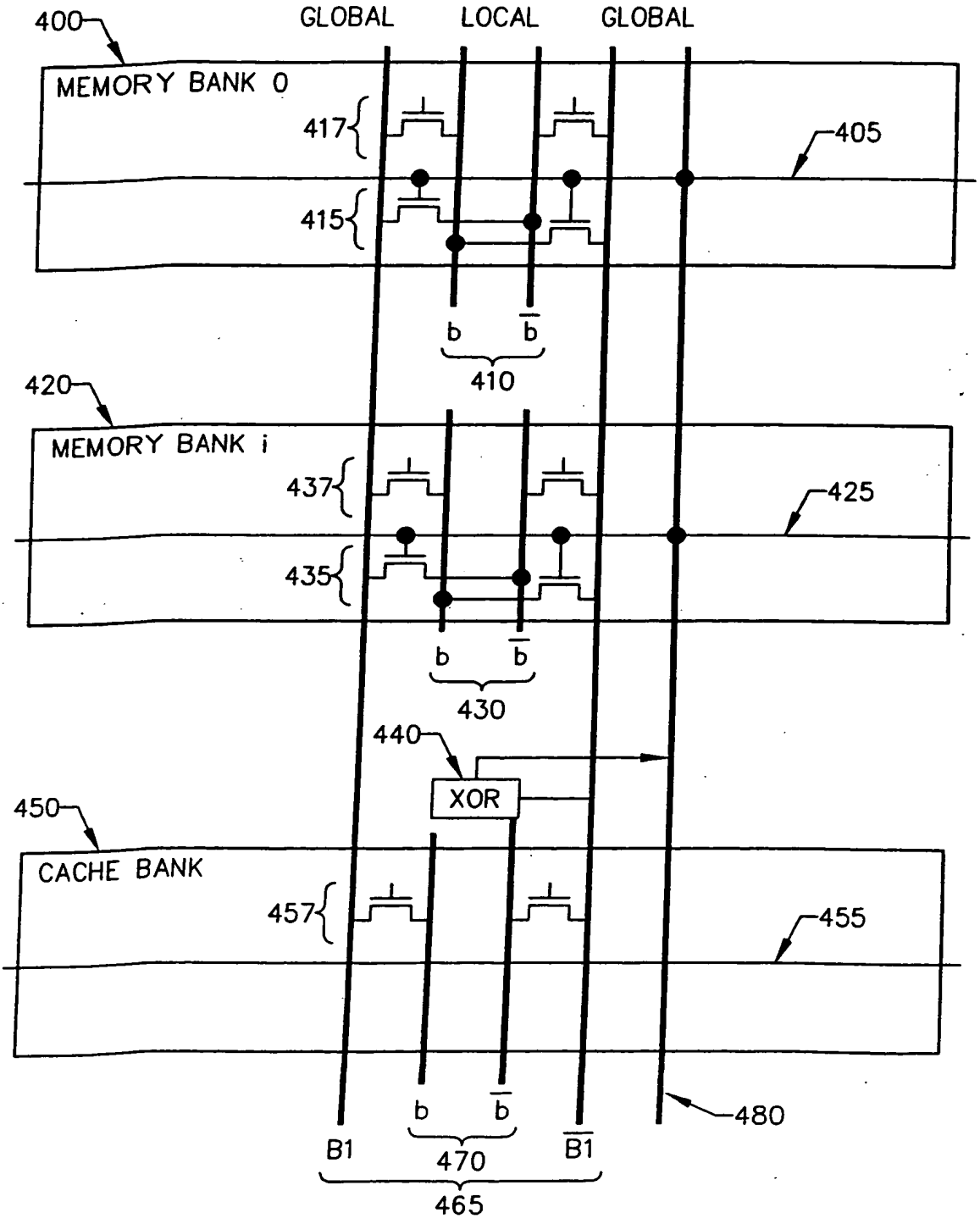

400
GLOBAL
LOCAL
GLOBAL
MEMORY BANK 0
417
415
405
b
b̄
410
420
MEMORY BANK i
437
435
425
b
b̄
430
440
XOR
450
CACHE BANK
457
455
480
b
b̄
B1
B̄1
470
465

FIGURE 5

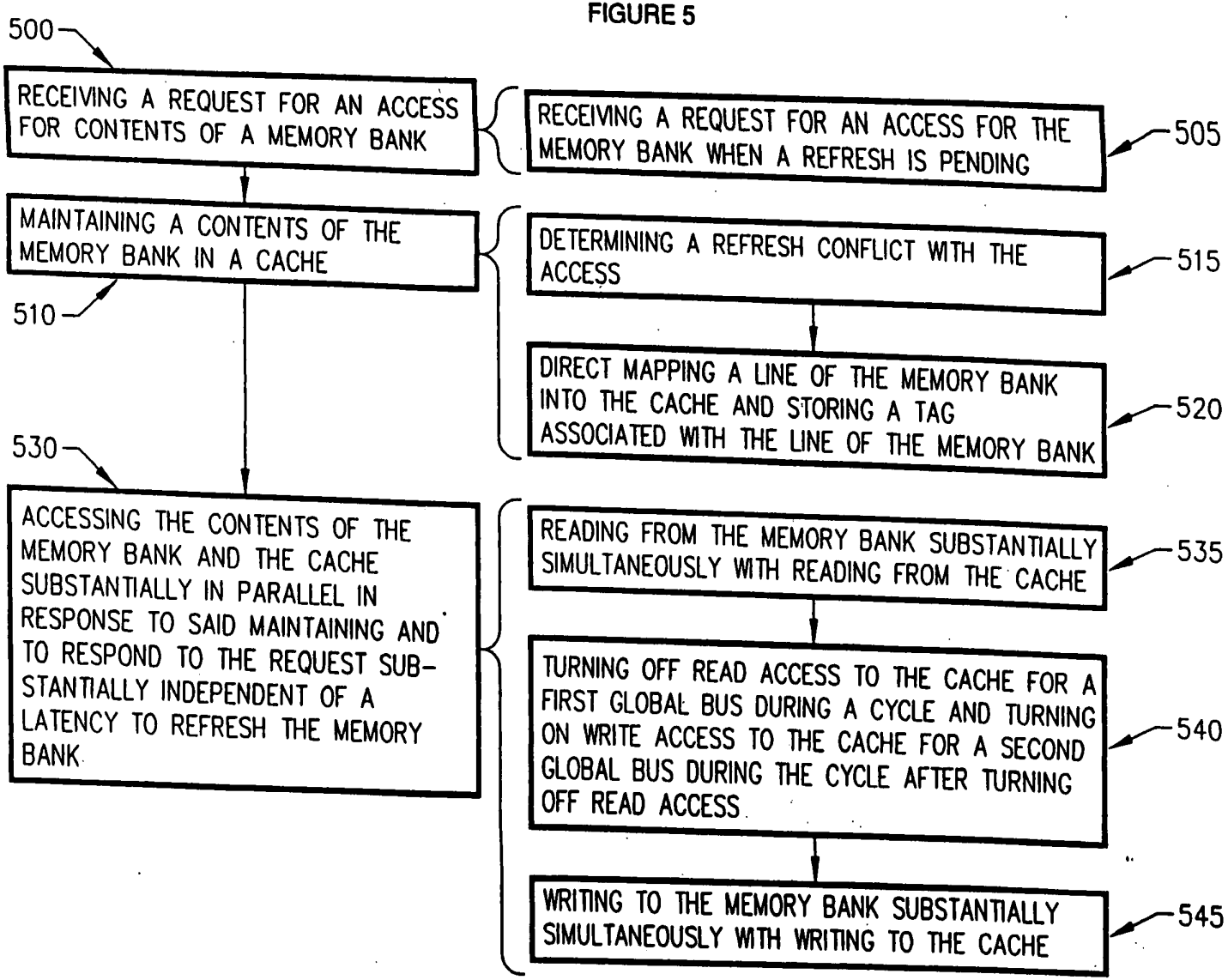
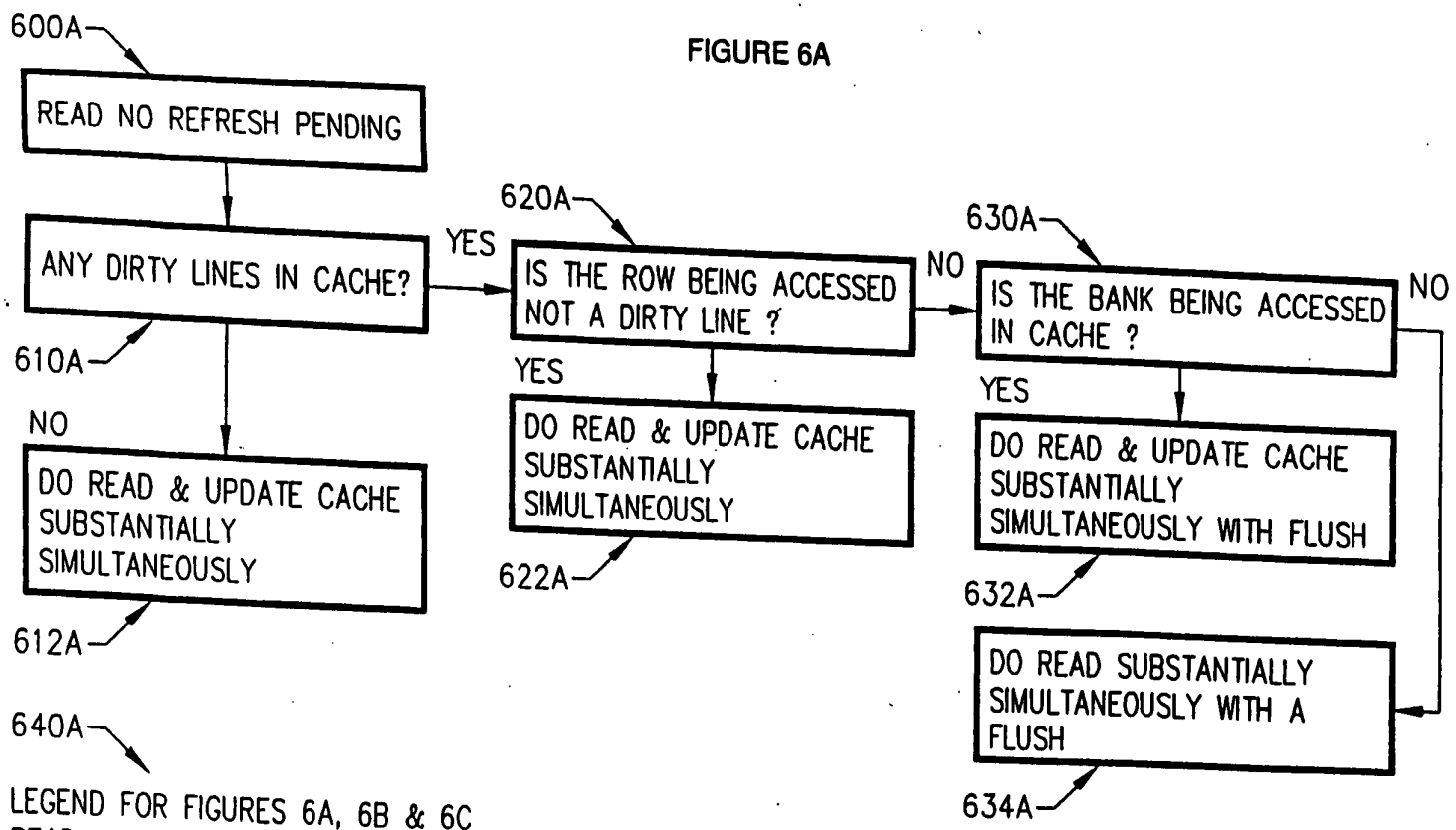

500
RECEIVING A REQUEST FOR AN ACCESS FOR CONTENTS OF A MEMORY BANK
RECEIVING A REQUEST FOR AN ACCESS FOR THE MEMORY BANK WHEN A REFRESH IS PENDING
505
MAINTAINING A CONTENTS OF THE MEMORY BANK IN A CACHE
510
DETERMINING A REFRESH CONFLICT WITH THE ACCESS
515
DIRECT MAPPING A LINE OF THE MEMORY BANK INTO THE CACHE AND STORING A TAG ASSOCIATED WITH THE LINE OF THE MEMORY BANK
520
530
ACCESSING THE CONTENTS OF THE MEMORY BANK AND THE CACHE SUBSTANTIALLY IN PARALLEL IN RESPONSE TO SAID MAINTAINING AND TO RESPOND TO THE REQUEST SUBSTANTIALLY INDEPENDENT OF A LATENCY TO REFRESH THE MEMORY BANK
READING FROM THE MEMORY BANK SUBSTANTIALLY SIMULTANEOUSLY WITH READING FROM THE CACHE
535
TURNING OFF READ ACCESS TO THE CACHE FOR A FIRST GLOBAL BUS DURING A CYCLE AND TURNING ON WRITE ACCESS TO THE CACHE FOR A SECOND GLOBAL BUS DURING THE CYCLE AFTER TURNING OFF READ ACCESS
540
WRITING TO THE MEMORY BANK SUBSTANTIALLY SIMULTANEOUSLY WITH WRITING TO THE CACHE
545

FIGURE 6A



640A

LEGEND FOR FIGURES 6A, 6B & 6C

READ = READ THE ROW BEING ACCESSED IN THE MEMORY BANK

READ CACHE = READ THE CACHE VERSION OF THE ROW BEING ACCESSED

UPDATE CACHE = WRITE THE ROW FROM THE MEMORY BANK BEING ACCESSED INTO THE CACHE AND SUBSTANTIALLY SIMULTANEOUSLY STORE A CACHE TAG FOR THE ENTRY

FLUSH = COPY THE CACHE VERSION OF A ROW INTO THE CORRESPONDING BANK SUBSTANTIALLY SIMULTANEOUSLY WITH CLEANING THE DIRTY BIT FOR THE ROW

REFRESH = REFRESH THE ROW MARKED AS REFRESH PENDING

BLOCK REFRESH = PREVENT THE REFRESH OF THE ROW UNTIL AN ACTION MAY BE COMPLETED

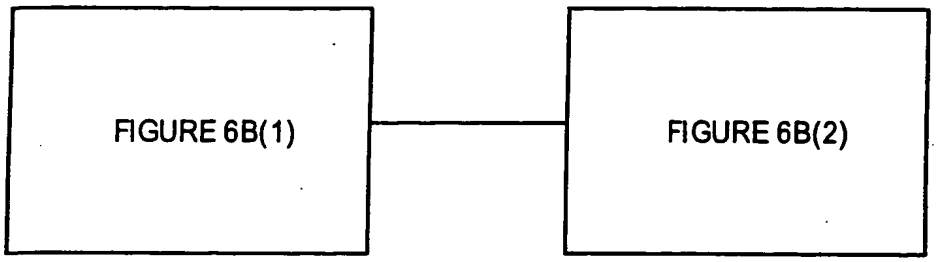


FIG. 6B

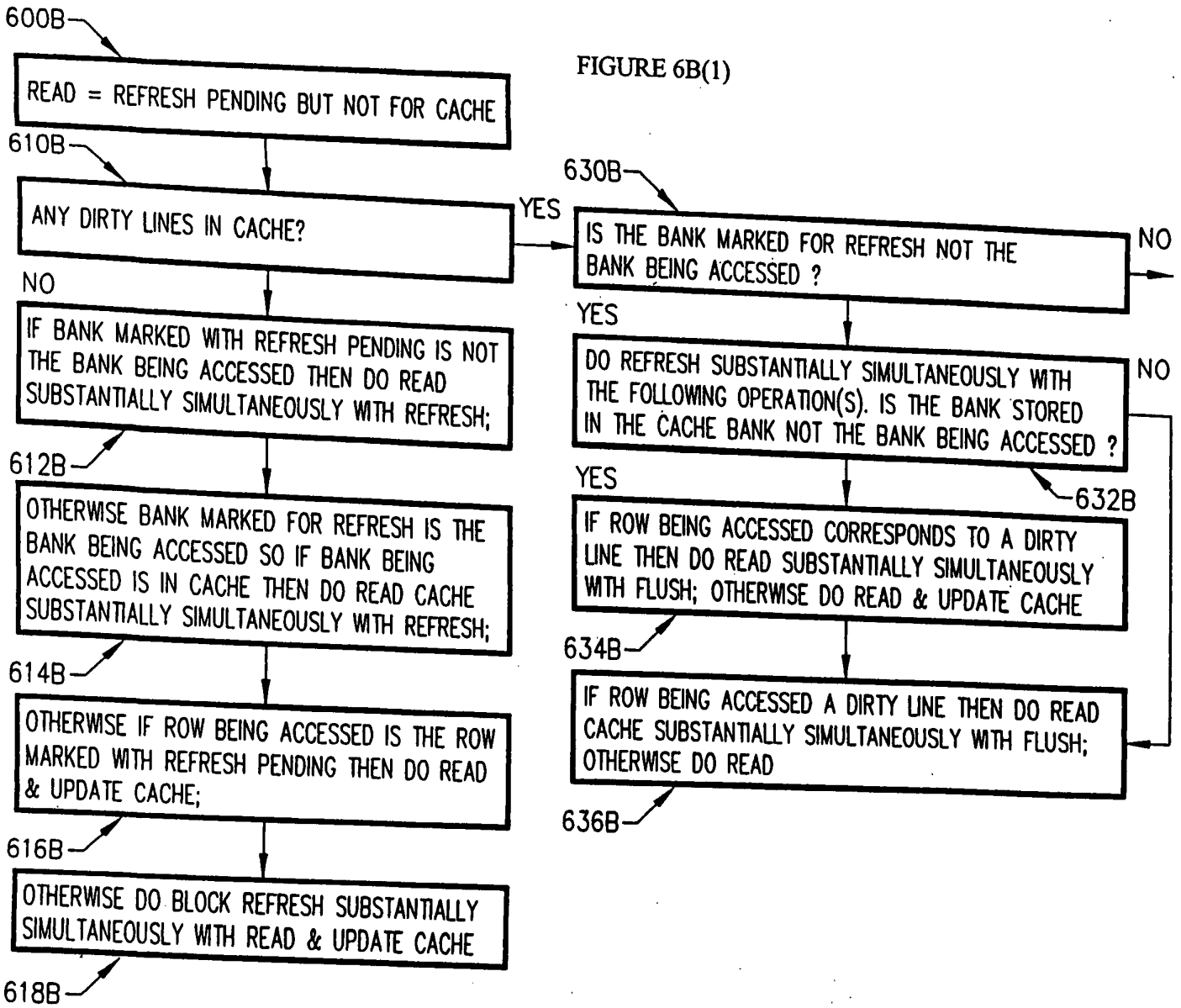

600B
READ = REFRESH PENDING BUT NOT FOR CACHE
610B
ANY DIRTY LINES IN CACHE?
YES
NO
IF BANK MARKED WITH REFRESH PENDING IS NOT THE BANK BEING ACCESSED THEN DO READ SUBSTANTIALLY SIMULTANEOUSLY WITH REFRESH;
612B
OTHERWISE BANK MARKED FOR REFRESH IS THE BANK BEING ACCESSED SO IF BANK BEING ACCESSED IS IN CACHE THEN DO READ CACHE SUBSTANTIALLY SIMULTANEOUSLY WITH REFRESH;
614B
OTHERWISE IF ROW BEING ACCESSED IS THE ROW MARKED WITH REFRESH PENDING THEN DO READ & UPDATE CACHE;
616B
OTHERWISE DO BLOCK REFRESH SUBSTANTIALLY SIMULTANEOUSLY WITH READ & UPDATE CACHE
618B
630B
IS THE BANK MARKED FOR REFRESH NOT THE BANK BEING ACCESSED ?
NO
YES
DO REFRESH SUBSTANTIALLY SIMULTANEOUSLY WITH THE FOLLOWING OPERATION(S). IS THE BANK STORED IN THE CACHE BANK NOT THE BANK BEING ACCESSED ?
NO
YES
632B
IF ROW BEING ACCESSED CORRESPONDS TO A DIRTY LINE THEN DO READ SUBSTANTIALLY SIMULTANEOUSLY WITH FLUSH; OTHERWISE DO READ & UPDATE CACHE
634B
IF ROW BEING ACCESSED A DIRTY LINE THEN DO READ CACHE SUBSTANTIALLY SIMULTANEOUSLY WITH FLUSH; OTHERWISE DO READ
636B

FIGURE 6B(1)

FIGURE 6B(2)

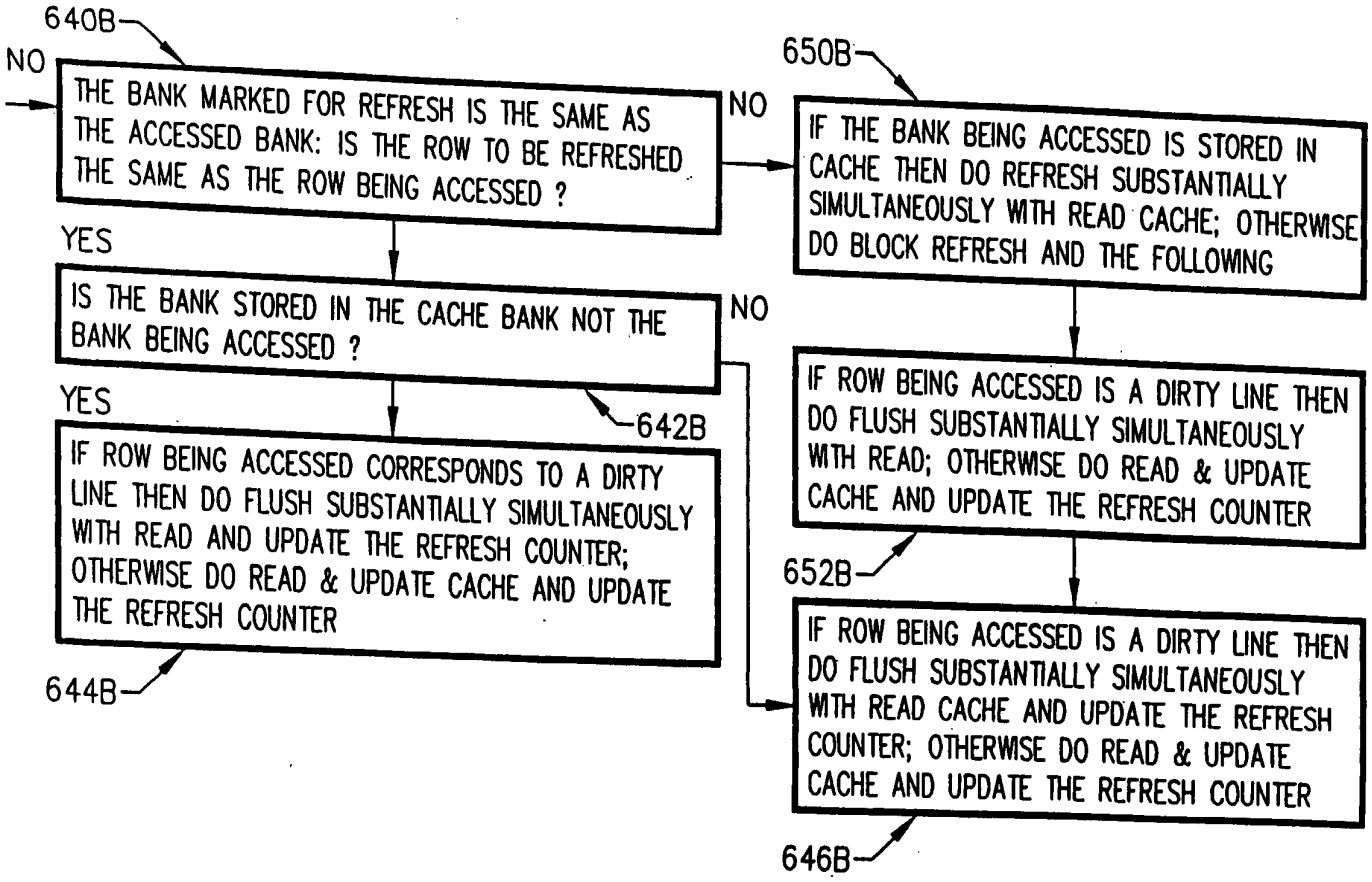

640B
NO
THE BANK MARKED FOR REFRESH IS THE SAME AS THE ACCESSED BANK: IS THE ROW TO BE REFRESHED THE SAME AS THE ROW BEING ACCESSED ?
YES
IS THE BANK STORED IN THE CACHE BANK NOT THE BANK BEING ACCESSED ?
642B
YES
IF ROW BEING ACCESSED CORRESPONDS TO A DIRTY LINE THEN DO FLUSH SUBSTANTIALLY SIMULTANEOUSLY WITH READ AND UPDATE THE REFRESH COUNTER; OTHERWISE DO READ & UPDATE CACHE AND UPDATE THE REFRESH COUNTER
644B
NO
650B
IF THE BANK BEING ACCESSED IS STORED IN CACHE THEN DO REFRESH SUBSTANTIALLY SIMULTANEOUSLY WITH READ CACHE; OTHERWISE DO BLOCK REFRESH AND THE FOLLOWING
IF ROW BEING ACCESSED IS A DIRTY LINE THEN DO FLUSH SUBSTANTIALLY SIMULTANEOUSLY WITH READ; OTHERWISE DO READ & UPDATE CACHE AND UPDATE THE REFRESH COUNTER
652B
IF ROW BEING ACCESSED IS A DIRTY LINE THEN DO FLUSH SUBSTANTIALLY SIMULTANEOUSLY WITH READ CACHE AND UPDATE THE REFRESH COUNTER; OTHERWISE DO READ & UPDATE CACHE AND UPDATE THE REFRESH COUNTER
646B

FIGURE 6C

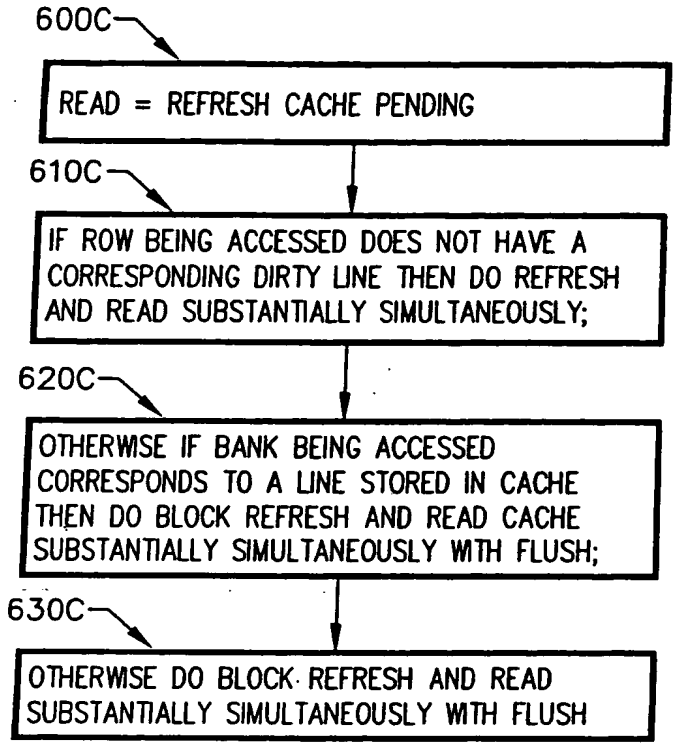
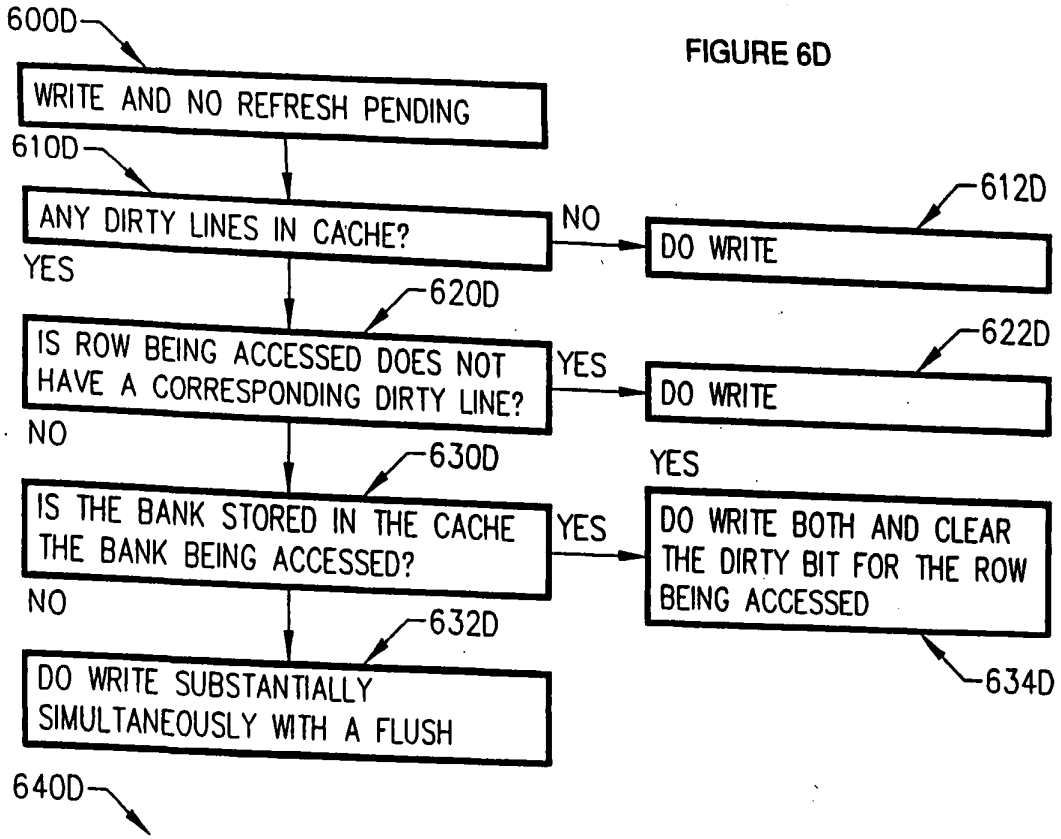

600C
READ = REFRESH CACHE PENDING
610C
IF ROW BEING ACCESSED DOES NOT HAVE A CORRESPONDING DIRTY LINE THEN DO REFRESH AND READ SUBSTANTIALLY SIMULTANEOUSLY;
620C
OTHERWISE IF BANK BEING ACCESSED CORRESPONDS TO A LINE STORED IN CACHE THEN DO BLOCK REFRESH AND READ CACHE SUBSTANTIALLY SIMULTANEOUSLY WITH FLUSH;
630C
OTHERWISE DO BLOCK REFRESH AND READ SUBSTANTIALLY SIMULTANEOUSLY WITH FLUSH

FIGURE 6D



LEGEND FOR FIGURES 6D, 6E & 6F

WRITE = WRITE TO THE ROW BEING ACCESSED IN THE MEMORY BANK

WRITE CACHE = WRITE TO THE CACHE VERSION OF THE ROW BEING ACCESSED

UPDATE CACHE = WRITE THE ROW FROM THE MEMORY BANK BEING ACCESSED INTO THE CACHE AND SUBSTANTIALLY SIMULTANEOUSLY STORE A CACHE TAG FOR THE ENTRY

FLUSH = COPY THE CACHE VERSION OF A ROW INTO THE CORRESPONDING BANK SUBSTANTIALLY SIMULTANEOUSLY WITH CLEANING THE DIRTY BIT FOR THE ROW

REFRESH = REFRESH THE ROW MARKED AS REFRESH PENDING

BLOCK REFRESH = PREVENT THE REFRESH OF THE ROW UNTIL AN ACTION MAY BE COMPLETED

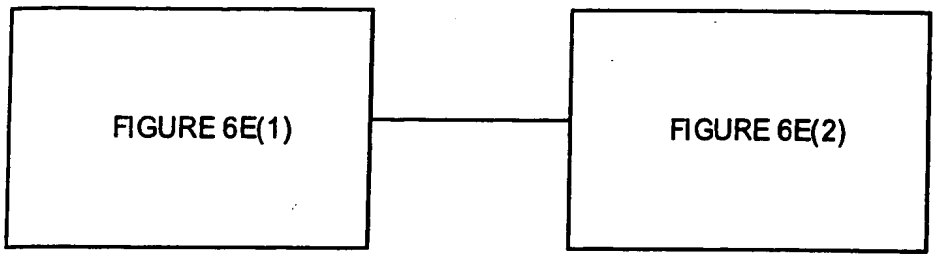


FIG. 6E

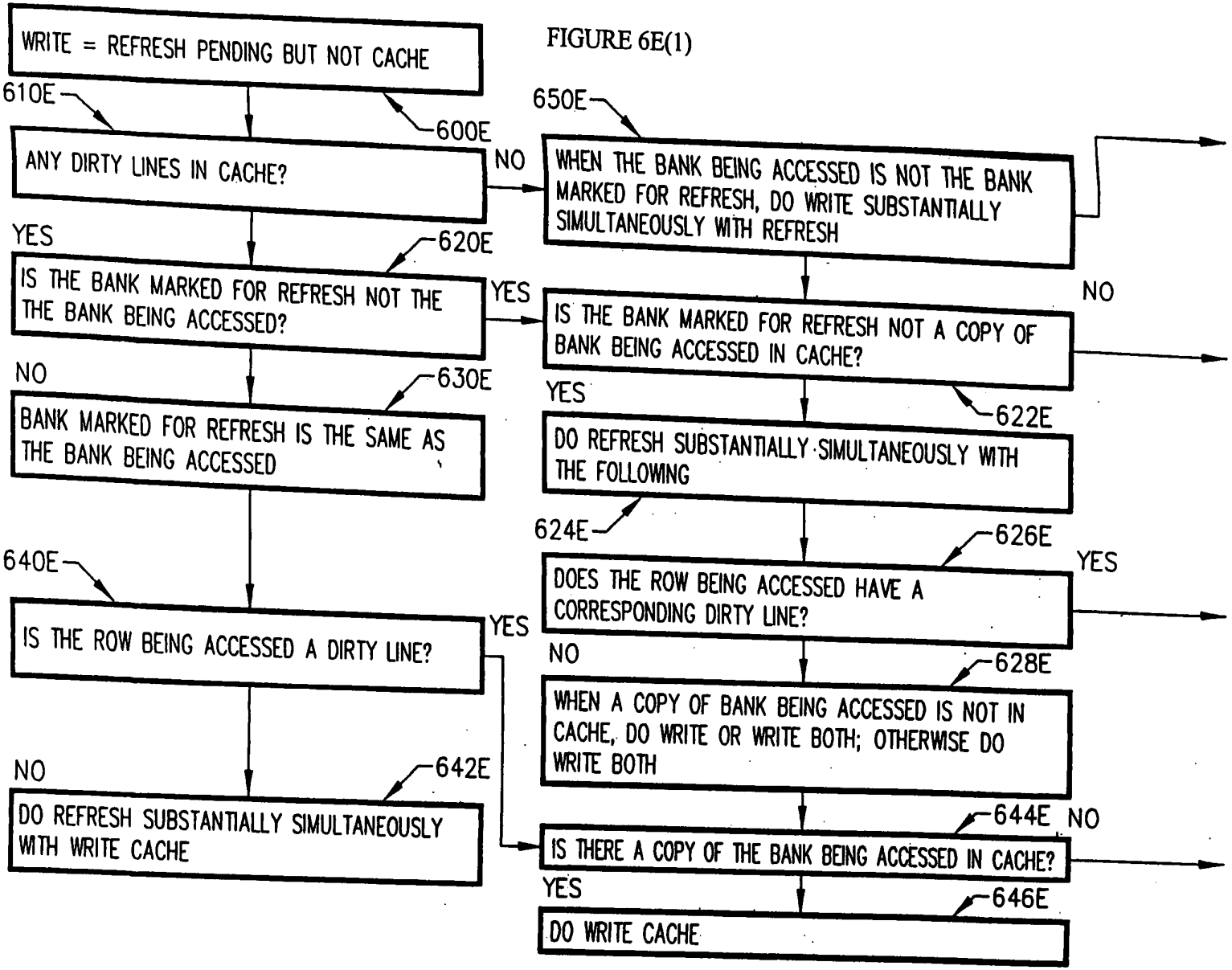

WRITE = REFRESH PENDING BUT NOT CACHE
610E
600E
ANY DIRTY LINES IN CACHE?
NO
650E
WHEN THE BANK BEING ACCESSED IS NOT THE BANK MARKED FOR REFRESH, DO WRITE SUBSTANTIALLY SIMULTANEOUSLY WITH REFRESH
YES
620E
IS THE BANK MARKED FOR REFRESH NOT THE THE BANK BEING ACCESSED?
YES
IS THE BANK MARKED FOR REFRESH NOT A COPY OF BANK BEING ACCESSED IN CACHE?
NO
NO
630E
BANK MARKED FOR REFRESH IS THE SAME AS THE BANK BEING ACCESSED
YES
622E
DO REFRESH SUBSTANTIALLY SIMULTANEOUSLY WITH THE FOLLOWING
624E
626E
640E
DOES THE ROW BEING ACCESSED HAVE A CORRESPONDING DIRTY LINE?
YES
IS THE ROW BEING ACCESSED A DIRTY LINE?
YES
NO
628E
WHEN A COPY OF BANK BEING ACCESSED IS NOT IN CACHE, DO WRITE OR WRITE BOTH; OTHERWISE DO WRITE BOTH
NO
642E
644E
NO
DO REFRESH SUBSTANTIALLY SIMULTANEOUSLY WITH WRITE CACHE
IS THERE A COPY OF THE BANK BEING ACCESSED IN CACHE?
YES
646E
DO WRITE CACHE

FIGURE 6E(1)

FIGURE 6E(2)

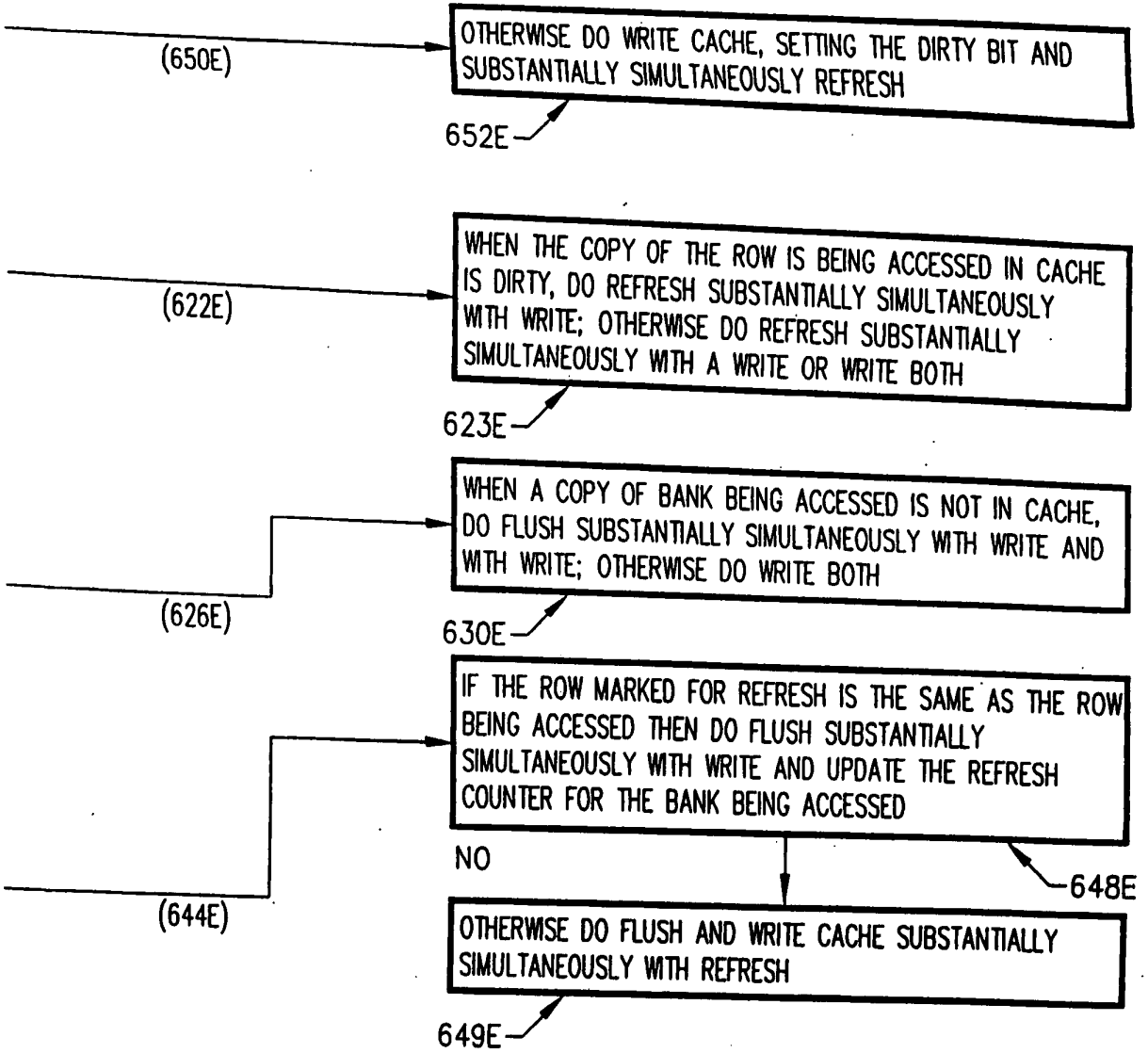

(650E)
OTHERWISE DO WRITE CACHE, SETTING THE DIRTY BIT AND SUBSTANTIALLY SIMULTANEOUSLY REFRESH
652E
(622E)
WHEN THE COPY OF THE ROW IS BEING ACCESSED IN CACHE IS DIRTY, DO REFRESH SUBSTANTIALLY SIMULTANEOUSLY WITH WRITE; OTHERWISE DO REFRESH SUBSTANTIALLY SIMULTANEOUSLY WITH A WRITE OR WRITE BOTH
623E
(626E)
WHEN A COPY OF BANK BEING ACCESSED IS NOT IN CACHE, DO FLUSH SUBSTANTIALLY SIMULTANEOUSLY WITH WRITE AND WITH WRITE; OTHERWISE DO WRITE BOTH
630E
(644E)
IF THE ROW MARKED FOR REFRESH IS THE SAME AS THE ROW BEING ACCESSED THEN DO FLUSH SUBSTANTIALLY SIMULTANEOUSLY WITH WRITE AND UPDATE THE REFRESH COUNTER FOR THE BANK BEING ACCESSED
NO
648E
OTHERWISE DO FLUSH AND WRITE CACHE SUBSTANTIALLY SIMULTANEOUSLY WITH REFRESH
649E

FIGURE 6F

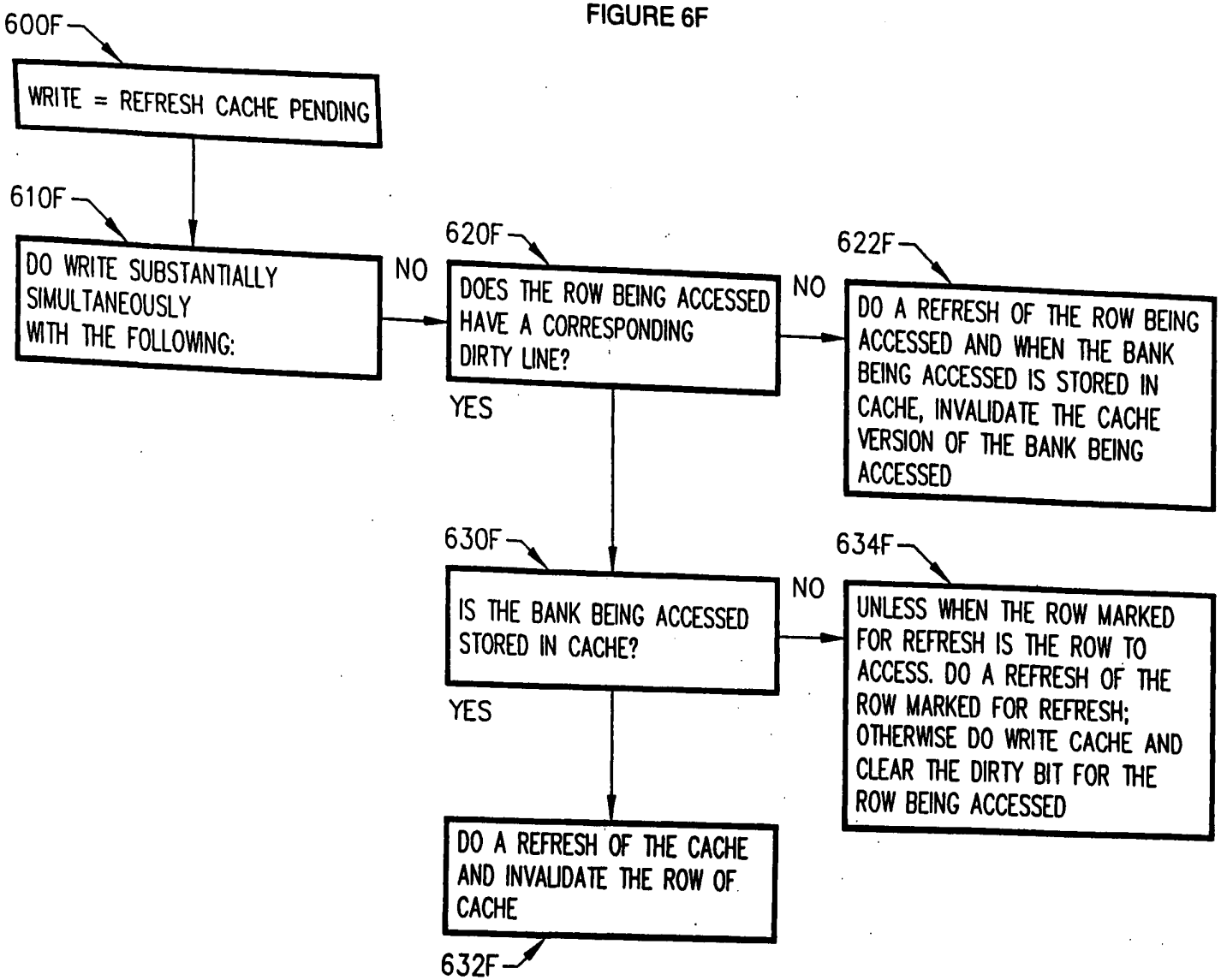

600F
WRITE = REFRESH CACHE PENDING
610F
DO WRITE SUBSTANTIALLY SIMULTANEOUSLY WITH THE FOLLOWING:
NO
620F
DOES THE ROW BEING ACCESSED HAVE A CORRESPONDING DIRTY LINE?
NO
622F
DO A REFRESH OF THE ROW BEING ACCESSED AND WHEN THE BANK BEING ACCESSED IS STORED IN CACHE, INVALIDATE THE CACHE VERSION OF THE BANK BEING ACCESSED
YES
630F
IS THE BANK BEING ACCESSED STORED IN CACHE?
NO
634F
UNLESS WHEN THE ROW MARKED FOR REFRESH IS THE ROW TO ACCESS. DO A REFRESH OF THE ROW MARKED FOR REFRESH; OTHERWISE DO WRITE CACHE AND CLEAR THE DIRTY BIT FOR THE ROW BEING ACCESSED
YES
DO A REFRESH OF THE CACHE AND INVALIDATE THE ROW OF CACHE
632F

FIGURE 7A

INPUTS

```
Refresh Counter:              rb, rr   /* refresh bank, refresh row */
Current Access Address:       wb, wr /* access (write) bank and row */
Number of rows in a bank: row
```

BUFFERS

```
Valid[row]                    /* valid bits for each of the rows in the cache bank*/

Dirty[row]                    /* dirty bits for cache lines*/
Cache[row]                    /* cache bank */
Cachetag[row]                 /* Tag for cache lines */
```

FUNCTIONS

```
Refresh()       {mem[rb, rr] = mem[rb, rr]; }
Flush()         {mem[cachetag[wr], wr] = cache[wr] $$ dirty[wr] = 0;}
                /* WB dirty line */
ReadUpdateC()       {data_out = mem[wb, wr] $$ cache[wr] = mem[wb, wr]
                    $$ cachetag[wr] = wb; }
Read()          {data_out = mem[wb, wr]; }
Write()         {mem[wb, wr] = data_in; }
ReadCache()  {data_out = cache[wr] $$ cache[wr] = cache[wr];}
WriteCache()  {cache[wr] = data_in $$ cachetag[wr] = wb $$ dirty[wr] = 1;}
WriteBoth()     {Write() $$ Writecache()$$ dirty[wr] = 0; }
                /* write to both the cache and memory */
```

($$ - means operations are done in parallel)

PSEUDO PROGRAM DESCRIPTION

```
/* Initialize all cachetags to be NULL */
Cachetag[*] = NULL;

/*** READ ***/

/*-------- No  refresh pending --------- */
if (no refresh pending) {
        if (no dirty line) {ReadUpdateC(); }
        else /* there is a dirty line */ {
                if (!dirty[wr]) { ReadUpdateC();}
                else if (wb == cachetag[wr]) { ReadCache() $$ Flush() $$ dirty[wr] = 0; }
                     else { Read() $$ Flush(); }
```

FIGURE 7B

```
/* ---------Refresh pending -------*/
else {
    If (rb != cache) {
        If (no dirty line) {
            if (rb != wb) { Read() $$ Refresh(); }
            else { /* rb == wb */
                if (wb == tag[wr])
                    { ReadCache() $$ Refresh(); }
                else {
                    if (wr == rr) ReadUpdateC();
                    else Block_refresh $$ ReadUpdateC();
                }
            }
        }
        else /* there is a dirty line */
            if (rb != wb) {
                Refresh() $$
                if (tag[wr] != wb) {
                    if (dirty[wr]) { Flush() $$ Read();}
                    else ReadUpdateC();
                }
                else {
                    if (dirty[wr]) { ReadCache() $$ Flush();}
                    else Read();
                }
            } /* end if rb != wb */
            else { /* refresh bank is the same as the accessed bank */
                if ( wr == rr) {
                    if (tag[wr] != wb) {
                        if (dirty[wr]) {Flush() $$ Read()
                          $$ UpdateRefreshCounter(); }
                        else {ReadUpdateC() $$ Update Refresh
                    Cntr();}
                    }
                    else {
                        if (dirty[wr]) { ReadCache() $$ Flush()
                            $$ UpdaterefreshCntr;}
                        else { ReadUpdateC() $$
                    UpdateRefreshCounter();}
                    }
                }
                else {
                    if (tag[wr] == wb) { Refresh() $$ ReadCache();}
                    /* dirty status for the line unchanged */
                    else {
                        BlockRefresh;
                        if (dirty[wr]) { Flush() $$ Read(); }
                        else ReadUpdateC();
                    }
                }
            }
    } /* end else there is a dirty line */
} /* end if rb != cache */
```

FIGURE 7C

```
        else /* the cache bank is being refreshed */ {
                if (dirty[wr]) {
                        if (tag[wr] == wb) {
                                BlockRefresh;
                                ReadCache() $$ Flush();
                        }
                        else {
                                BlockRefresh;
                                Read() $$ Flush();
                        }
                }
                else {
                        Refresh()
                        Read()
                }
        }
} /* end refresh pending */
```

FIGURE 7D

```
/*** WRITE ***/

/* ------ No pending refresh ------- */
if (no refresh pending) {
        /* there is no dirty line */
        if (no dirty line) {Write(); }
        else {  /* there is a dirty line */
                if (!dirty[wr])  { Write();}
                else {
                        if (Cachetag[wr] == wb) { WriteBoth() $$ dirty[wr] = 0; }
                        else {Write() $$ Flush();}
                }
        } /* end dirty line */
}
```

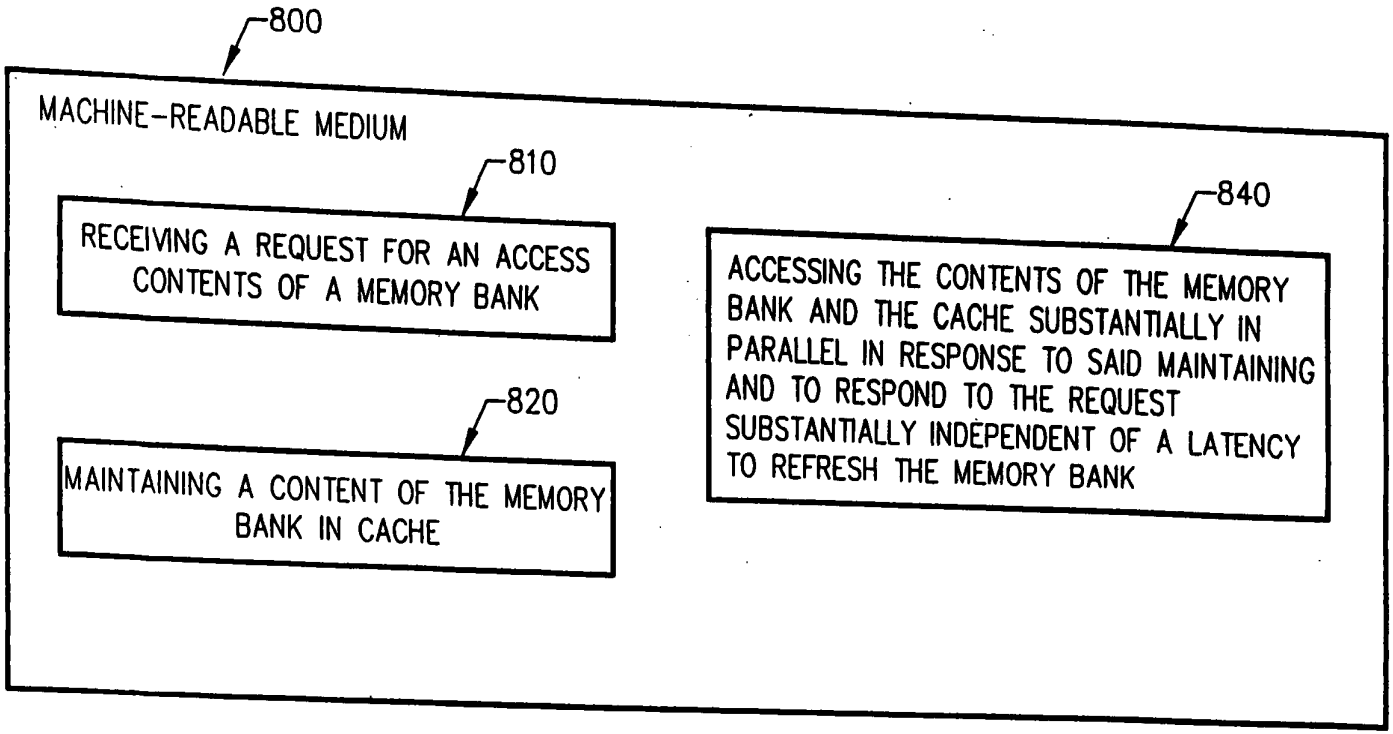
FIGURE 7E

```
/* ------ Pending refresh ------- */
else {
    if (rb != cache) {
      if (no dirty line) {
            if (wb != rb) { Write() $$ Refresh() }
            else { WriteCache() $$ dirty[wr] = 1 $$ Refresh() ;}
      }
      else {  /* there is a dirty line */
            if (rb != wb) { /* refresh bank is different from the access bank */
              if (rb != cachetag[wr]) {
                  Refresh() $$
                  If (dirty[wr]) {
                        If (cachetag[wr] != wb) {Flush() $$ Write();}
                        Else {Writeboth();}
                  }
                  else {
                        If (cachetag[wr] != wb) { Write();} /* or writeboth() */
                        Else {Writeboth();}
                  }
              else {
                  if (dirty[wr]) { Refresh() $$ Write();} /* cache line remains dirty */
                  else { Refresh() $$ Write() ;} /* Or writeboth() */
              }
            }
            else { /* refresh bank is the same as the access bank */
                  If (dirty[wr]) {
                        If (cachetag[wr] == wb) { Writecache();}
                        else {
                              if (wr == rr) {/* lucky day*/
                                    Flush() $$ write() $$ Updaterefreshcounter;
                              }
                              else {
                                    Flush() and write to cache with new data ;
                                    $$ Refresh();
                              }
                        }
                  else { /* the line is not dirty */
                  /* create another dirty line */
                        Refresh $$
                        WriteCache();
                  }
            }
    }
```

FIGURE 7F

```
else { /* cache bank is being refreshed */
        Write() $$
        if (dirty[wr]) {
                if (cachetag[wr]  == wb) {
                        Refresh() $$
                        Cachetag[wr] = NULL;
                }
                else {
                        Refresh();
                        /* unless wr = rr then we writeback and clear dirty bit */
                }
        }
        else { /* not dirty */
                Refresh();
                if (cachetag[wr]  == wb) { Cachetag[wr] = NULL;}
        }
    }
}
```

FIGURE 8


800
MACHINE-READABLE MEDIUM
810
RECEIVING A REQUEST FOR AN ACCESS CONTENTS OF A MEMORY BANK
820
MAINTAINING A CONTENT OF THE MEMORY BANK IN CACHE
840
ACCESSING THE CONTENTS OF THE MEMORY BANK AND THE CACHE SUBSTANTIALLY IN PARALLEL IN RESPONSE TO SAID MAINTAINING AND TO RESPOND TO THE REQUEST SUBSTANTIALLY INDEPENDENT OF A LATENCY TO REFRESH THE MEMORY BANK